INF-253 Lenguajes de Programación

Tarea 4: Scheme

16 de octubre de 2023

1. Cooking Programma'

La famosa ama de casa *Cooking Programma'* ha decidido abrir un restaurante para que todo el mundo pruebe su deliciosa comida. El local es un éxito, quizás demasiado; debido a lo popular que es *Programma'* todos quieren probar su comida, la chef se encuentra superada por la cantidad de pedidos que recibe cada día, olvidando por completo ir a comprar ingredientes e incluso perdiendo algunos utensilios de la cocina.

Aquí es donde entra usted, luego de presenciar la enorme fila al local, decide ayudar a su vecina creando un programa en Scheme que la ayude a alivianar algunas de sus tareas.

2. Funciones a Implementar

1. **Checkear Cantidad**
   - **Sinopsis:** `(checkear cantidad lista)`
   - **Característica Funcional:** Funciones simples, listas simples, recursión simple.
   - **Descripción:** Se le entrega una *cantidad* que dice la supuesta longitud de *lista*. La función checkear debe retornar si *cantidad* coincide con la longitud de *lista*.
     NOTA: Solo se pueden usar las funciones básicas de listas: first, rest, cons, list y append. **No se permite el uso de length, debe hacerse con recursión sobre la lista.**
   - **Ejemplos:**

```
> (checkear 4 '(a b c d))
true
> (checkear 2 ‘(a a c d e w w q t a v))
false
> (checkear 0 ‘())
true
```

2. **Calcular Proporción**
   - **Sinopsis:** `(cantidades_cola base lista)` y `(cantidades_simple base lista)`
   - **Característica Funcional:** Funciones lambda, recursión simple, recursión de cola
   - **Descripción:** A partir de un número base y una lista de funciones *f(x)*, se debe retornar una lista donde cada elemento es *f(base)*. Las funciones *f(x)* solo tendrán + - * / como operaciones a realizar. La implementación decir considerar dos funciones, una con recursión simple y otra con recursión con cola.
   - **Ejemplos:**

```
> (cantidades_cola 2 (list (lambda (x) (/ x 2)) (lambda (x) (* x 3)) (lambda (x) (- x 2))))
(1 6 0)

> (cantidades_cola 2 (list (lambda (x) (/ (+ x 2) 2)) (lambda (x) (* x 4)) (lambda (x) (* (/ x 3) 2))))
```

(2 8 $1\frac{1}{3}$)

3. **Lista de Compras**
   - **Sinopsis:** (armar_lista stock)
   - **Característica Funcional:** Funciones simples, listas anidadas, recursión simple
   - **Descripción:** Basándose en lo aprendido en **Checkear Cantidad** (no use length), ahora se necesita armar una lista de compras para obtener los ingredientes que faltan. Se entrega un *stock*, una lista con pares *(cant_necesaria lista)* donde *lista* es una lista homogénea de elementos que denota cuántos ingredientes de un tipo hay, y *cant_necesaria* es un entero que indica cuánto se necesita de ese ingrediente. La función debe retornar una lista de pares *(cant_comprar ingrediente)*, que expresa cuánto hay que comprar del ingrediente para alcanzar la *cant_necesaria*.
   - **Ejemplo:**

```
> (armar_lista '((5 (cebolla cebolla)) (3 (tomate tomate tomate)) (2 (ajo))))

((3 cebolla) (1 ajo))
```

NOTA: Solo para este ejercicio, no se entregarán inputs donde la lista de ingredientes esté vacía. También, si la cantidad a comprar es 0, no se incluye en el retorno.

4. **Buscar Receta**
   - **Sinopsis:** `(buscar_recetas ingrediente lista)`
   - **Característica Funcional:** Funciones puras, listas anidadas, recursión simple
   - **Descripción:** Debe retornar todos los nombres de las recetas que contienen el *ingrediente*, donde *lista* sigue la siguiente estructura:

     ```
     ((nombre_receta1 ingrediente1 ingrediente2 …) (nombre_receta2 …))
     ```

   - **Ejemplos:**

```
> (buscar_recetas '(salmon) '((pescado_arroz salmon atun) (ratatouille
tomate berenjena romero) (sushi salmon arroz wasabi) (cazuela papa carne
zapallo apio)))

(pescado_arroz sushi)

> (buscar_recetas '() '((pescado_frito arroz atun) (ratatouille queso
fideos pimenton) (sushi atun salmon arroz palta) (cazuela papa carne
espinaca choclo)))

()

>(buscar_recetas '(arroz) '())

()
```

NOTA: Debe implementar su propia versión de *ingrediente in lista*, es decir, deben recorrer recursivamente cada receta buscando el ingrediente. (usen `eqv?` para verificar igualdad de ingredientes)

5. **¿Dónde dejé los vasos?**
   - **Sinopsis** `(buscar_utensilio utensilio arbol)`
   - **Característica Funcional:** Funciones simples, listas anidadas (estructura de árbol), recursión simple
   - **Descripción:** Debe encontrar dónde está un *utensilio* dentro de la cocina, donde los muebles y objetos son representados por un árbol binario de la siguiente forma: `(valor_nodo árbol_izquierdo árbol_derecho)`
     Una vez encuentre el utensilio, debe recorrer el camino que siguió en el *arbol* para llegar a él. Si el utensilio se encuentra en múltiples lugares, debe retornar el camino más rápido a él (si hay 2 caminos igual de rápidos se retorna uno arbitrariamente).
   - **Ejemplos:**

```
> (buscar_utensilio '(vaso) '(cocina (arriba (alacena (gabinete () (vaso
() ())) (vaso () ())) (estante (tenedor () ()) (batidor () ()))) (abajo
(taburete (vaso () ()) ()) (cajon (batidor () ()) (tenedor () ()))))))

(cocina arriba alacena)

> (buscar_utensilio ‘() ‘())

“no está”
```

IMPORTANTE: Todos los inputs entregados serán correctos, es decir, no tiene que realizar revisión de casos bordes, aunque sí debe identificar inputs vacíos. Esto implica que no ocurrirán divisiones por 0, por ejemplo.

3. Sobre la Entrega

- Se deberá entregar un archivo por cada ejercicio con la(s) función(es) solicitadas, con el siguiente formato de nombres:
    - P1.rkt, P2.rkt, P3.rkt, P4.rkt, P5.rkt
- Se debe programar siguiendo el paradigma de la programación funcional, no realizar códigos que siguen el paradigma imperativo. Por ejemplo, se prohíbe el uso de for-each. Para implementar las funciones utilice DrRacket.
    - http://racket-lang.org/download/
- Todo código debe contener al principio `#lang scheme`
- Todos los archivos deben ser de la extensión **.rkt**
- Pueden crear funciones que no estén especificadas para utilizar en los problemas planteados, pero solo se revisará que la función pedida funcione y el problema esté resuelto con las características funcionales planteada en el enunciado.
- Cuidado con el orden y la indentación de su tarea, llevará descuento de lo más 20 puntos.
- Las funciones implementadas y que no estén en el enunciado deben ser comentadas de la siguiente forma. **SE HARÁN DESCUENTOS POR FUNCI´ON NO COMENTADA**

```
; ; Descripción de la función
; ;
; ; a : Descripción del parámetro a
; ; b : Descripción del parámetro b
```

- Se debe trabajar de forma individual obligatoriamente.
- La entrega debe realizarse en tar.gz y debe llevar el nombre: Tarea4LP_RolAlumno.tar.gz. Ejemplo: *Tarea4LP_202200000-0.tar.gz*
- El archivo README.txt debe contener nombre y rol del alumno e instrucciones detalladas para la correcta utilización de su programa. De no incluir README se realizará un descuento de 20 puntos.
- La entrega será vía aula y el plazo máximo de entrega es hasta el martes 31 de octubre a las 23:55.
- Las copias serán evaluadas con nota 0 y se informarán a las respectivas autoridades.
- **Solo se responderán consultas sobre la Tarea hasta 24 horas antes de la fecha de entrega.**

4. Calificación

9.1. Entrega

Para la calificación de su tarea, debe realizar una entrega con requerimientos mínimos que otorgarán 30 pts base, luego se entregará puntaje dependiendo de los otros requerimientos que llegue a cumplir.

9.1.1. Entrega Mínima

- Implementar la función 1 (**Checkear Cantidad**). (10 pts)
- Implementar las dos versiones de recursión para la función 2 (**Calcular Proporción**). (20 pts, 10 pts por cada tipo de recursión)

NOTA: Las funciones deben entregar respuestas correctas en el formato denotado en sus ejemplos.

9.1.2. Entrega

Luego de cumplir con la Entrega Mínima, puede obtener más puntaje cumpliendo con los siguientes puntos (puede haber puntaje parcial por cada punto):

- Implementación de funciones para la correcta resolución de los casos de prueba (Total 70 pts)
    - Implementar **Lista de Compras** (15 pts)
    - Implementar **Buscar Receta** (25 pts)
    - Implementar **¿Dónde dejé los vasos?** (30 pts)

Para todas las funciones, existe puntaje parcial acorde a los casos de prueba que resuelve.

9.2 Descuentos

Luego de cumplir con la Entrega Mínima, puede sufrir descuentos por los siguientes motivos:

- Falta de comentarios (-5 pts c/u Max 20 pts)
- Falta de README (-20 pts)
- Falta de alguna información obligatoria en el README (-5 pts c/u)
- Falta de orden (entre -5 y -20 pts dependiendo de que tan desordenado)
- Entrega tardía (-20 pts por día, -10 pts dentro de la primera hora)
- Mal nombre en algún archivo entregado (-5 pts c/u, -20 pts si es el .tar.gz)